Model-Free Reinforcement Learning for Lexicographic Omega-Regular Objectives*

Ernst Moritz Hahn[1], Mateo Perez[2], Sven Schewe[3], Fabio Somenzi[2], Ashutosh Trivedi[2], and Dominik Wojtczak[3]

[1] University of Twente, The Netherlands
[2] University of Colorado Boulder, USA
[3] University of Liverpool, UK

Abstract. We study the problem of finding optimal strategies in Markov decision processes with lexicographic ω-regular objectives, which are ordered collections of ordinary ω-regular objectives. The goal is to compute strategies that maximise the probability of satisfaction of the first ω-regular objective; subject to that, the strategy should also maximise the probability of satisfaction of the second ω-regular objective; then the third and so forth. For instance, one may want to guarantee critical requirements first, functional ones second and only then focus on the non-functional ones. We show how to harness the classic off-the-shelf model-free reinforcement learning techniques to solve this problem and evaluate their performance on four case studies.

1 Introduction

In the basic setting of model-free reinforcement learning (RL), the goal is to find a strategy that optimises the total accumulated (discounted) reward, given a single reward structure. However, this setting is seldom sufficient to concisely express real-world scenarios. On the one hand, it is often necessary to consider objectives for which it is not obvious how they could be expressed by a reward structure. On the other hand, in many cases multiple objectives are of interest, and not all of them are equally important.

This paper studies model-free RL for the lexicographic optimization of ω-regular objectives. That is, we are given k different ω-regular specifications, ordered by importance. We then optimise the probabilities of these specifications in the following way. An optimal strategy has to maximise the probability that first objective is achieved. The strategy then also has to maximise the probability that second objective holds among those strategies that achieve the maximum probability for the first objective. The next priority is then to maximise the third objective, and so forth.

* This work was supported by the Engineering and Physical Sciences Research Council through grant EP/P020909/1 and by the National Science Foundation through grant 2009022. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No 101032464 (SyGaST), 864075 (CAESAR), and 956123 (FOCETA).

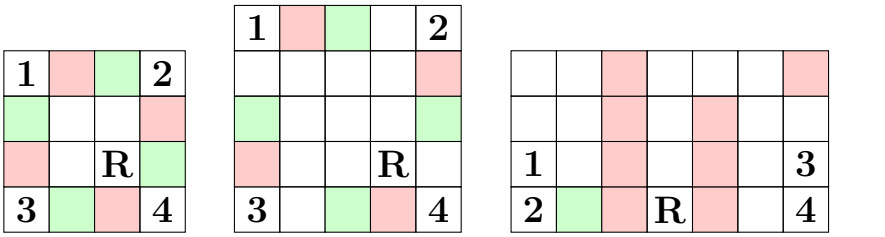

Fig. 1: Robot case study: grid-world examples with multiple ω-regular objectives.

Consider a robot that moves around in a grid world. We depict three such scenarios of different sizes in Figure 1. In each step, the robot can choose to go into one of four directions. If there is enough space, the robot might move one or two steps: With probability $\mathbf{p}$, the robot moves two steps into the chosen direction, while it moves only a single step into the selected direction with a probability of $1-\mathbf{p}$. If there is only room for one step, the robot deterministically moves one step. If there is no room for a step, then the robot doesn't move.

In the figure, the robot's initial position is marked with **R**. Red fields are dangerous for the robot, while the green ones allow the robot to recharge its battery. Four fields are marked with **1** to **4**. For these scenarios (a)–(c), we have three different objectives expressed as LTL formulas. Among them 1. marks the objective with the highest priority, while 3. marks the objective with the lowest.

In scenario (a), the first priority of the robot is to visit all four numbered fields infinitely often, the second priority is never to enter a dangerous area, and the third priority is to visit some charging field infinitely often. In this scenario, it is possible to fulfil the primary objective with probability 1. However, the secondary objective can then not be fulfilled with any probability larger than 0: Due to its uncertain movements, the robot always has a chance to step into a dangerous field when trying to fulfil the primary objective. The third priority, however, can again be fulfilled with probability 1.

The objectives of scenario (b) are the same as for (a). However, because of the larger grid, the robot can stay out of danger while satisfying its primary objective; it can meet all three objectives with probability 1.

In scenario (c), the primary objective is to visit fields **1** and **2** infinitely often *or* to visit fields **3** and **4** infinitely often. The secondary objective is to avoid dangerous fields. The tertiary one is to eventually visit a charging field. Here, the robot can again fulfil its primary objective with probability 1. The secondary one can only be fulfilled with probability $1 - (\mathbf{p} \cdot (1 - \mathbf{p}))$: To fulfil its primary objective and then the secondary one, the best the robot can do is to move upwards and then to the right. If, by moving to the right, the robot first moves one field but then two fields, it will run into a dangerous field. When maximising the primary and secondary objectives, the tertiary objective can then only be

fulfilled with probability $\mathbf{p} \cdot (1 - \mathbf{p})$, because trying to reach the charging field cannot be done without entering a dangerous field with at least probability $1 - \mathbf{p}$ (the chance of not "jumping" over the left barrier of red fields). Therefore, the tertiary objective is only pursued when the secondary objective has been missed.

The priorities play a key role here: For scenario (c), if the safety objective $\mathbf{G}\neg$danger were the most important, the robot would be able to completely avoid unsafe fields, fulfilling this objective with probability 1. However, the probability to fulfil the recurrence objective $(\mathbf{GF}1 \wedge \mathbf{GF}2) \vee (\mathbf{GF}3 \wedge \mathbf{GF}4)$ would be $\mathbf{p}$, because the robot can only reach the lower-right corner of the grid with probability $\mathbf{p}$ if it is to avoid danger at all cost.

Our Approach. Our approach is based on the following idea. We assume that each objective is given as a Good-for-MDPs (GFM) Büchi automaton [12]. We compose our model with the product of these Büchi automata. We obtain a Markov decision process (MDP) whose transitions are labeled with multiple separate Büchi conditions. Then, generalizing a method described in previous work [13], we transform different combinations of Büchi conditions into different rewards in a reduction to a weighted reachability problem that depends on the prioritisation. We end up with an MDP equipped with a standard scalar reward, to which general RL algorithms can be applied.

Naturally, in practice, we do not perform the composition of the MDP with the automata and the transformation into rewards before the RL process starts; indeed, the two steps are intertwined, such that we perform the composition on-the-fly, avoiding the costly construction of parts of the product automaton that would never be visited during RL.

Related Work. The study of the optimal control problems for MDPs under various performance objectives is the subject of [20]. For a given MDP and performance objective (total reward, discounted reward, and average reward), the optimal expected cost can be characterised using Bellman equations and an optimal strategy can be computed using dynamic programming (value iteration or policy iteration) or linear programming [20]. Chatterjee, Majumadar, and Henzinger [6] considered MDPs with multiple discounted reward objectives. In the presence of multiple objectives, the trade-off between different objectives can be characterised as Pareto curves. The authors of [6] showed that every Pareto-optimal point can be achieved by a memoryless strategy and the Pareto curve can be approximated in polynomial time. Moreover, the problem of checking the existence of a strategy that realises a value vector can be decided in polynomial time. These multi-objective optimization problems were studied in the context of multiple long-run average objectives by Chatterjee [5]. He showed that the Pareto curve can be approximated in polynomial time in the size of the MDP for irreducible MDPs and in polynomial space in the size of the MDP for general MDPs. Additionally, the problem of checking the existence of a strategy that guarantees values for different objectives to be equal to a given vector is in polynomial time for irreducible MDPs and in NP for general MDPs.

Verification of stochastic systems against ω-regular requirements has received considerable attention [17,2]. For systems modeled as MDPs and requirements expressed using ω-regular specifications, the key verification problem "probabilistic model-checking" is to compute optimal satisfaction probabilities and strategies. The probabilistic model checking problem can be solved [1] using graph-theoretic techniques (by computing so-called accepting end-component and then maximising the probability to reach states in such components) over the product of MDPs and ω-automata. Etessami et al. [7] were the first to study the multi-objective model-checking problem for MDPs with ω-regular objectives. Given probability intervals for the satisfaction of various properties, they developed a polynomial-time (in the size of the MDP) algorithm to decide the existence of such a strategy. They also showed that, in general, such strategies may require both randomization and memory. That paper also studies the approximation of the Pareto curve with respect to a set of ω-regular properties in time polynomial in the size of the MDP. Forejt et al. [9] studied quantitative multi-objective optimization over MDPs that combines ω-regular and quantitative objectives. Those algorithms are implemented in the probabilistic model checker PRISM [17].

RL is concerned with the optimal control of MDPs when the transition table and the reward structures are not known to the agents, but can be learned by interacting with the environment. In such a case, the aforementioned dynamic programming solutions are not applicable. RL [23] provides a framework to learn optimal strategies from repeated interactions with the environment. There are two main approaches to RL in MDPs: *model-free* approaches and *model-based* approaches. In a model-based approach, the learner interacts with the system to first estimate the transition probabilities and corresponding rewards, and then uses dynamic programming algorithms to compute optimal values and strategies. On the other hand, model-free RL [22] refers to a class of techniques that are asymptotically space-efficient because they do not construct a full model of the environment. These techniques include classic algorithms like Q-learning [24] as well as their extensions that use neural networks, like deep Q-learning [19].

RL has recently been applied to finding optimal control for ω-regular objectives [21,10,16,11,14,15,12,4], but all of theses papers deal with a single objective. Recently [3], the problem of maximising the probability of satisfying a safety condition together with an single ω-regular objective was considered and, as a secondary objective, the controller aims at maximising a discounted reward. Our approach is more flexible than [3] as we do not require that safety objectives be prioritised over liveness objectives.

2 Preliminaries

Nondeterministic Büchi Automata. A *nondeterministic Büchi automaton* is a tuple $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \Gamma \rangle$, where Σ is a finite *alphabet*, Q is a finite set of *states*, $q_0 \in Q$ is the *initial state*, $\Delta \subseteq Q \times \Sigma \times Q$ is the set of transitions, and $\Gamma \subseteq Q \times \Sigma \times Q$ is the transition-based *acceptance condition*.

A *run* r of $\mathcal{A}$ on $w \in \Sigma^\omega$ is an ω-word $r_0, w_0, r_1, w_1, \ldots$ in $(Q \times \Sigma)^\omega$ such that $r_0 = q_0$ and, for $i > 0$, it is $(r_{i-1}, w_{i-1}, r_i) \in \Delta$. We write $\mathsf{inf}(r)$ for the set of transitions that appear infinitely often in the run r. A run r of $\mathcal{A}$ is *accepting* if $\mathsf{inf}(r) \cap \Gamma \neq \varnothing$. The *language*, $L_\mathcal{A}$, of $\mathcal{A}$ (or, *recognised* by $\mathcal{A}$) is the subset of words in Σ^ω that have accepting runs in $\mathcal{A}$. A language is ω-*regular* if it is accepted by a Büchi automaton.

An automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \Gamma \rangle$ is *deterministic* if $(q, \sigma, q'), (q, \sigma, q'') \in \Delta$ implies $q' = q''$ and is *complete* if, for all $\sigma \in \Sigma$ and $q \in Q$, there is a transition $(q, \sigma, q') \in \Delta$. A word has exactly one run in a deterministic, complete automaton.

Markov Decision Processes. A *Markov decision process (MDP)* $\mathcal{M}$ is a tuple $\langle S, s_0, A, T, \Sigma, L \rangle$ where S is a finite set of states, s_0 is a designated initial state, A is a finite set of *actions*, $T : S \times A \to \mathcal{D}(S)$, where $\mathcal{D}(S)$ is the set of probability distributions over S, is the *probabilistic transition (partial) function*, Σ is an alphabet, and $L : S \times A \times S \to \Sigma$ is the *labeling function* of the set of transitions. For a state $s \in S$, $A(s)$ denotes the set of actions available in s. For states $s, s' \in S$ and $a \in A(s)$, we have that $T(s, a)(s')$ equals $\Pr(s'|s, a)$.

A *run* of $\mathcal{M}$ is an ω-word $s_0, a_1, \ldots \in S \times (A \times S)^\omega$ such that $\Pr(s_{i+1}|s_i, a_{i+1}) > 0$ for all $i \geqslant 0$. A finite run is a finite such sequence. For a *run* $r = s_0, a_1, s_1, \ldots$ we define the corresponding labeled run as $L(r) = L(s_0, a_1, s_1), L(s_1, a_2, s_2), \ldots \in \Sigma^\omega$. We write $\mathit{Runs}(\mathcal{M})$ ($\mathit{FRuns}(\mathcal{M})$) for the set of runs (finite runs) of $\mathcal{M}$ and $\mathit{Runs}_s(\mathcal{M})$ ($\mathit{FRuns}_s(\mathcal{M})$) for the set of runs (finite runs) of $\mathcal{M}$ starting from state s. When the MDP is clear from the context we drop the argument $\mathcal{M}$.

A strategy in $\mathcal{M}$ is a function $\mu : \mathit{FRuns} \to \mathcal{D}(A)$ such that for all finite runs r we have $\mathit{supp}(\mu(r)) \subseteq A(\mathsf{last}(r))$, where $\mathit{supp}(d)$ is the support of d and $\mathsf{last}(r)$ is the last state of r. Let $\mathit{Runs}^\mu_s(\mathcal{M})$ denote the subset of runs $\mathit{Runs}_s(\mathcal{M})$ that correspond to strategy μ and initial state s. We say that a strategy μ is: *pure* if $\mu(r)$ assigns probability 1 to just one action in $A(\mathsf{last}(r))$ for all runs $r \in \mathit{FRuns}$; *stationary* if $\mathsf{last}(r) = \mathsf{last}(r')$ implies $\mu(r) = \mu(r')$ for all finite runs $r, r' \in \mathit{FRuns}$; and *finite-state* if there exists an equivalence relation $\sim$ on FRuns with a finite index, such that $\mu(r) = \mu(r')$ for all finite runs $r \sim r'$.

The behavior of an MDP $\mathcal{M}$ under a strategy μ with starting state s is defined on a probability space $(\mathit{Runs}^\mu_s, \mathcal{F}^\mu_s, \Pr^\mu_s)$ over the set of infinite runs of μ from s. Given a random variable over the set of infinite runs $f : \mathit{Runs} \to \mathbb{R}$, we write $\mathbb{E}^\mu_s\{f\}$ for the expectation of f over the runs of $\mathcal{M}$ from state s that follow strategy μ. A *Markov chain* is an MDP whose set of actions is a singleton. For any MDP $\mathcal{M}$ and stationary strategy μ, let $\mathcal{M}_\mu$ be the Markov chain resulting from choosing the actions in $\mathcal{M}$ according to μ.

An *end-component* of an MDP is a set $C \subseteq S$ such that for every $s \in C$ we can pick an action $a_s \in A(s)$ in such a way that $\{s' \mid T(s, a_s)(s') > 0\} \subseteq C$ and the graph with vertices in C and edges in $E = \{(s, s') \mid s \in C \text{ and } T(s, a_s)(s') > 0\}$ is strongly connected. An end-component is *accepting* if at least one of such edges correspond to an accepting transition and is *maximal* if there does not exist an end-component $C' \supseteq C$. It is well-known (see, e.g, [1]) that for every strategy the union of the end-components is visited with probability 1 and once

an end-component, C', is entered there is a pure finite-state strategy that visits its every edge infinitely many times while never leaving C'.

Syntactic and Semantic Satisfaction. Given an MDP $\mathcal{M}$ and an automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \Gamma \rangle$, we want to compute an optimal strategy satisfying the objective that the run of $\mathcal{M}$ is in the language of $\mathcal{A}$. We define the *semantic satisfaction* probability for $\mathcal{A}$ and a strategy μ from state s as:

$$\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s,\mu) = \Pr{}^{\mu}_{s}\big\{r \in Runs^{\mu}_{s}(\mathcal{M}) : L(r) \in L_{\mathcal{A}}\big\} \text{ and}$$
$$\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s) = \sup_{\mu}\big(\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s,\mu)\big).$$

A strategy μ_* is optimal for $\mathcal{A}$ if $\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s,\mu_*) = \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s)$.

When using automata for the analysis of MDPs, we need a syntactic variant of the acceptance condition. Given an MDP $\mathcal{M} = \langle S, s_0, A, T, \Sigma, L \rangle$ and an automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \Gamma \rangle$, the *product* $\mathcal{M} \times \mathcal{A} = \langle S \times Q, (s_0, q_0), A \times Q, T^{\times}, \Gamma^{\times} \rangle$ is an MDP augmented with an initial state (s_0, q_0) and accepting transitions $\Gamma^{\times}$. The function $T^{\times} : (S \times Q) \times (A \times Q) \rightharpoonup \mathcal{D}(S \times Q)$ is defined by

$$T^{\times}((s,q),(a,q'))((s',q')) = \begin{cases} T(s,a)(s') & \text{if } (q, L(s,a,s'), q') \in \Delta \\ 0 & \text{otherwise.} \end{cases}$$

Finally, $\Gamma^{\times} \subseteq (S \times Q) \times (A \times Q) \times (S \times Q)$ is defined by $((s,q),(a,q'),(s',q')) \in \Gamma^{\times}$ if, and only if, $(q, L(s,a,s'), q') \in \Gamma$ and $T(s,a)(s') > 0$. A strategy $\mu^{\times}$ on the product defines a strategy μ on the MDP with the same value, and vice versa. Note that for a stationary $\mu^{\times}$, the strategy μ may need memory. We define the *syntactic satisfaction* probabilities as

$$\mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}((s,q),\mu^{\times}) = \Pr{}^{\mu}_{s}\big\{r \in Runs^{\mu^{\times}}_{(s,q)}(\mathcal{M} \times \mathcal{A}) : \inf(r) \cap \Gamma^{\times} \neq \varnothing\big\}$$
$$\mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}(s) = \sup_{\mu^{\times}}\big(\mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}((s,q_0),\mu^{\times})\big) \ .$$

Note that $\mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}(s) = \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s)$ holds for a deterministic $\mathcal{A}$. In general, $\mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}(s) \leqslant \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s)$ holds, but equality is not guaranteed because the optimal resolution of nondeterministic choices may require access to future events.

An automaton $\mathcal{A}$ is *good for MDPs* (GFM), if $\mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}(s_0) = \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s_0)$ holds for all MDPs $\mathcal{M}$ [12]. For an automaton to match $\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s_0)$, its nondeterminism is restricted not to rely heavily on the future; rather, it must be possible to resolve the nondeterminism on-the-fly. In this paper we only consider GFM automata, which have this ability. Note that every LTL property and, more generally, every ω-regular objective can be expressed as a suitable GFM automaton [12].

For ω-regular objectives, optimal satisfaction probabilities and strategies can be computed using graph-theoretic techniques over the product structure. However, when the MDP transition structure is unknown, such techniques are not applicable. Model-free reinforcement learning overcomes this limitation.

3 From Lexicographic Objectives to Learning in 3.5 Steps

In this section, we provide a reduction from lexicographic ω-regular objectives to a payoff function (with hyperparameter) that can be wrapped for Q-learning (the additional half-step) in three steps. We assume that all properties are provided as good-for-MDP automata [12] based on which we construct an extended product automaton $\mathcal{P}$ in Subsection 3.4.

3.1 Lexicographic ω-regular Objectives

For two k-dimensional vectors $v = (v_1, \ldots, v_k)$ and $v' = (v'_1, \ldots, v'_k)$, we say that v is larger in the *lexicographic order* than v', denoted by $v > v'$, if there exists $1 \leqslant i \leqslant k$ such that $v_i > v'_i$ and $v_j = v'_j$ for all $j < i$. We write $v \geqslant v'$ if $v > v'$ or $v = v'$. The problem we address is the following:

> *Given MDP $\mathcal{M}$ with unknown transition structure and k GFM Büchi automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$ accepting ω-regular objectives $\varphi_1, \ldots, \varphi_k$, compute a strategy optimal for the* lexicographic ω-regular objective $(\mathcal{A}_1, \ldots, \mathcal{A}_k)$*, that is, a strategy that maximises according to the lexicographic order the vector $(p_1, \ldots, p_k)$ where $p_i = \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}_i}(s_0)$ is the probability that $\mathcal{M}$ satisfies φ_i.*

We adapt the model-free reinforcement learning (RL) framework to solve this problem. Bridging the gap between ω-regular specifications and model-free RL requires a translation from specifications to scalar rewards, such that a model-free RL algorithm maximising scalar rewards produces a strategy that maximises the probability to satisfy the specification. In this section we show how this can also be done for lexicographic ω-regular objectives.

3.2 Optimal Strategies for Lexicographic Objectives

Before turning to the reduction, we consider optimal strategies for the product of the MDP $\mathcal{M}$ and the automaton $\mathcal{P}$ that combines all the individual objectives. We also recapitulate what we need to achieve in $\mathcal{M} \times \mathcal{P}$ and what we can assume about optimal strategies (such that some optimal strategies will satisfy these assumptions).

Broadly speaking, the goal for each individual Büchi objective is to reach an end-component with an accepting transition for this objective, and to stay in this end-component forever, realizing this Büchi condition. maximising the probability of reaching such an end-component is the key objective, while, in the end-component, the strategy needs to make sure that some accepting transition is (almost surely) visited infinitely often.

For Büchi objectives with a lexicographic order, the reachability of a "good" end-component is the part that becomes lexicographic: the main objective is to maximise the chance of reaching an end-component, where the main objective is satisfied. Among all strategies that achieve this, we then maximise the chance of satisfying the secondary objective, and so forth.

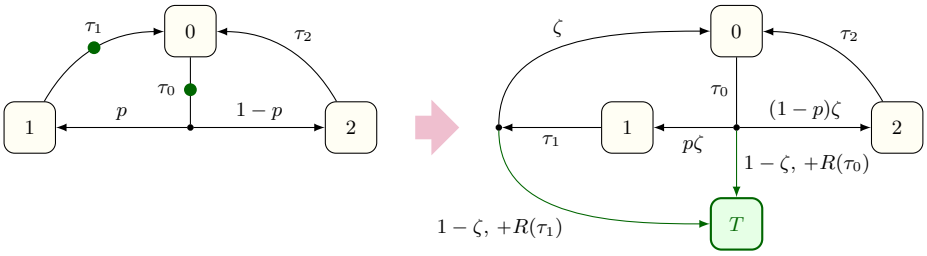

Fig. 2: Adding transitions to the target (T) in the augmented product MDP. The only edges that give non-zero reward are marked with a transition dependent $+R(\cdot)$ reward. In the original translation [11], all such rewards were equal to 1.

As we have Büchi objectives, it is in principle possible to only consider maximal end-components for this. While this cannot be a learning objective—as there is neither a concept of maximality in reinforcement learning, nor is learning such solutions a natural goal—it means that, whenever a state of $\mathcal{M} \times \mathcal{P}$ is in two end-components that satisfy the main and the secondary Büchi condition, then it is also in an end-component that satisfies both—for example, the union of the former two end-components. This prevents the interplay between these goals from becoming complex.

Once we have reached an end-component in $\mathcal{M} \times \mathcal{P}$ we want to stay in, the principle strategy is again to cover it, and thus to satisfy all objectives that this end-component satisfies almost surely.

3.3 Outline of Our Reduction

The reduction generalises the reduction from [11] for ω-regular objectives to reachability. The difference in the reduction is that it uses a *weighted* extension of reachability, i.e., different transitions to the target may be assigned different rewards. These weights are determined by the transition taken by an extended product automaton, which takes into account which set of individual objectives have recently been progressed towards by traversing accepting transitions of their corresponding GFM Büchi automata.

Broadly speaking, if we used this approach directly with a lexicographic order, it would translate a positional strategy that satisfies the individual properties with probabilities $(p_1, p_2, \ldots, p_k)$ to a "reachability" vector $(p'_1, p'_2, \ldots, p'_k)$, such that $p_i \leqslant p'_i \leqslant p_i + \varepsilon$, holds, where ε is bounded from above by $(1-\zeta) \cdot E$, where ζ is the parameter from the reduction (Figure 2), and E is the maximal (among pure positional strategies) expected number of accepting transitions visited before reaching an end-component that the strategy remains in.

Note that one cannot simply optimise the lexicographic value of $(p'_1, p'_2, \ldots, p'_k)$, as the noise in the assessment of the probability to meet the main objective, no matter how small, would outweigh all real differences in the probability of meeting the less relevant objectives. However, as the set of positional strategies is finite, there is a minimal difference $p_{\min} > 0$ between the probabilities to

achieve an individual objective by any two positional strategies with a different probability of meeting this objective.

Let $f_k = 1$, pick any $f_i \geqslant (1 + 1/p_{\min})f_{i+1}$ for all $i < k$, and let $\overrightarrow{f} = (f_1, f_2, \ldots, f_k)$. Now, consider any two (pure and stationary) strategies that obtain probability vectors of satisfying the properties $\overrightarrow{p} = (p_1, p_2, \ldots, p_k)$ and $\overrightarrow{p}' = (p'_1, p'_2, \ldots, p'_k)$, respectively, such that $\overrightarrow{p} > \overrightarrow{p}'$. We claim that the value of $\overrightarrow{p} \cdot \overrightarrow{f}^T$ is at least $p_{\min}$ higher than $\overrightarrow{p}' \cdot \overrightarrow{f}^T$. This is because, assuming that the i-th position is the first one where $\overrightarrow{p}$ and $\overrightarrow{p}'$ differ, we get:

$$\begin{aligned}
\overrightarrow{p} \cdot \overrightarrow{f}^T - \overrightarrow{p}' \cdot \overrightarrow{f}^T &\geqslant p_{\min} f_i - \sum_{j>i} f_j \\
&\geqslant p_{\min}(1 + 1/p_{\min}) f_{i+1} - \sum_{j>i} f_j \\
&= p_{\min} f_{i+1} - \sum_{j>i+1} f_j \geqslant \ldots \geqslant p_{\min} f_k = p_{\min}.
\end{aligned}$$

Moreover, if $\sum_{i=1}^{k} f_i \cdot \varepsilon < p_{\min}$, then using the weights of $\overrightarrow{f}$ for the reachability will guarantee that a strategy with a better performance obtains a better value and, in particular, only optimal strategies can obtain the highest value.

3.4 From Lexicographic Objectives to Lexicographic Büchi

In a first step, we discuss reductions from lexicographic ω-regular objectives, given as GFM Büchi automata, to GFM automaton with lexicographic Büchi objectives. In what follows, let $\mathbb{B} = \{0, 1\}$ be the set of Boolean values.

Lexicographic Büchi Objectives. We first have to give semantics for lexicographic Büchi objectives. For convenience in the proofs, we provide two: the *independent* semantics and the *infimum* semantics. We will show that we obtain the correct results, irrespective of which semantics we use, for the translations we suggest: both semantics provide the same value for the extended product automaton we define.

For this, we extend the usual product automaton by equipping the 'accepting transitions' Γ with a valuation function $v : \Gamma \to \mathbb{B}^k$, which maps each accepting transition to a k dimensional Boolean (0/1) vector different from the $\vec{0}$ vector, where each dimension refers to one of the k individual Büchi objectives.

The *infimum* semantics assigns to any run, r, the value according to the 'worst accepting transition' seen infinitely many times in r. Namely, to the set $I_A = \mathsf{inf}(r) \cap \Gamma$ of accepting transitions visited infinitely many times along r, it assigns the $\vec{0}$ vector if I_A is empty, and otherwise the lexicographically minimal vector in $\{v(t) \mid t \in I_A\}$.

The *independent* semantics intuitively treats all Büchi conditions independently: using it, a run would be assigned a Boolean vector $(b_1, b_2, \ldots, b_k)$, where $b_i = 1$ if there is a transition $t \in I_A$ such that the i^{th} component of $v(t)$ is 1.

Reductions. We assume that the individual ω-regular objectives are given by k GFM Büchi automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$, with $\mathcal{A}_i = (\Sigma, Q^i, q_0^i, \Delta^i, \Gamma^i)$. From these automata, we discuss a construction to an equivalent extended product automaton $\mathcal{P}$, where equivalence means that, for all finite-state strategies, the value that can be obtained using the k GFM automata for the properties, and the infimum and independent semantics for the two automata we construct provide the same results. Our construction builds an automaton $\mathcal{P} = (\Sigma, Q, q_0, \Delta, \Gamma, v)$, where

- $Q = \times_{i=1}^{k}(Q^i \times \mathbb{B})$
- $q_0 = (q_0^1, 0; q_0^2, 0; \ldots; q_0^k, 0)$
- $\Delta = \Gamma \cup \Delta'$, where the non-accepting transitions Δ' are defined independently for the k components: for the i^{th} component,
  - $((q,0), \sigma, (q',0))$ is possible iff $(q, \sigma, q') \in \Delta^i \backslash \Gamma^i$ (i.e., iff (q, σ, q') is a non-accepting transition of $\mathcal{A}_i$),
  - $((q,0), \sigma, (q',1))$ is possible iff $(q, \sigma, q') \in \Gamma^i$ (i.e. iff, (q, σ, q') is an accepting transition of $\mathcal{A}_i$),
  - $((q,1), \sigma, (q',1))$ is possible iff $(q, \sigma, q') \in \Delta^i$ (i.e., iff (q, σ, q') is a transition of $\mathcal{A}_i$), and
- for all transitions $(q_1, b_1; \ldots; q_k, b_k; \sigma; q_1', b_1'; \ldots; q_k', b_k') \in \Delta'$ with $\sum_{i=1}^{k} b_k' \neq 0$, Γ contains a transition $t = (q_1, b_1; \ldots; q_k, b_k; \sigma; q_1', 0; \ldots; q_k', 0)$ (obtained by replacing all Boolean values in the target state by 0), with $v(t) = (b_1', b_2', \ldots, b_k')$; Γ contains no further transitions.

That is, Δ' simply collects the information, which of the individual accepting transitions has been seen since the last transition from Γ has been taken. Taking a transition from Γ then 'cashes in' on these transitions, while resetting the tracked values to 0.

As a minor optimisation, we remove the states $\times_{i=0}^{k-1} Q^i \times \{1\}$ together with the transitions that lead to them. This can be done as there is never a point in delaying to cash in on these transitions. Offering such additional choices that should never be taken would likely impede, rather than help, learning.

Equivalence. Recalling that pure finite-memory strategies suffice for MDPs with lexicographic Büchi objectives (follows from [7]), we now show equivalence.

Theorem 1. *Given an MDP $\mathcal{M}$ and k ω-regular objectives given as GFM Büchi automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$, it holds that maximising these k objectives with lexicographic order, and maximising them with the automaton $\mathcal{P}$ from above with infimum or independent semantics provides the same result.*

Proof. We make use of the fact that pure finite-memory strategies suffice to obtain optimal control. We therefore fix an arbitrary finite-memory strategy μ for the control of the MDP $\mathcal{M}$ with lexicographic Büchi objectives, obtaining a Markov chain $\mathcal{M}_\mu$. Note that this is only a control for the MDP, not of the witness automaton $\mathcal{P}$.

We first turn any pure strategy for $\mathcal{P}$ with independent semantics into a strategy for the individual $\mathcal{A}_i$, which yields the same expected vector for every

run (and thus the same expected probability vector). This is quite simple: every individual $\mathcal{A}_i$ can simply behave like its component in $\mathcal{P}$. On every run, if the i^{th} component of the lexicographic vector is 1, then $\mathcal{A}_i$ has seen infinitely many accepting transitions.

Next, we turn k individual pure finite-memory strategies for the individual $\mathcal{A}_i$ into a strategy for $\mathcal{P}$ and evaluate it with the infimum semantics. For this, the automaton essentially follows the component strategies, and only has to additionally decide when to 'cash in.' $\mathcal{P}$ will make this choice whenever all individual automata have reached an end-component on the product of $\mathcal{M}_\mu$ and the automaton and, for all $\mathcal{A}_i$ that are in an accepting end-component, the Boolean store in the i^{th} component is set to 1, or would have been set in this move. Note that this means that $v(t)$ in this case indicates all those components, for whom the individual $\mathcal{A}_i$ is in an accepting end-component whenever an accepting transition occurs. (In case that none of the $\mathcal{A}_i$ is in an accepting end-component, we do not use accepting transitions.)

Apart from this, it only uses accepting transitions when all Boolean values would otherwise be 1 (because then cashing in is forced).

With this strategy, the valuation can only differ from the individual valuations of the $\mathcal{A}_i$ if either (at least) one of the $\mathcal{A}_i$-s never reaches an end-component, or if it eventually reaches an accepting end-component, but only visits accepting transitions finitely often. As both of these events have probability 0, the expected vectors are the same for the individual $\mathcal{A}_i$ and for $\mathcal{P}$ with this strategy.

Finally, for every strategy the expected value for the independent semantics is at least as high as the value for the infimum semantics. □

3.5 From Lexicographic Büchi to Weighted Büchi

We first observe that the previous theorem translates smoothly to a scalar version of the previous translation: we call a Büchi automaton *weighted* if it has a positive weight function $w : \Gamma \to \mathbb{R}$ instead of the lexicographic value function v. Similar to lexicographic Büchi, the value of a run is then 0 if no accepting transition occurs infinitely often. If accepting transitions do occur infinitely often, and they do occur in the order $t_1, t_2, t_3, \ldots \in \Gamma^\omega$, than the value of the run is $\liminf_{n\to\infty}(1/n)\sum_{i=1}^{n} w(t_i)$.

For any given linear function $f : \mathbb{R}^k \to \mathbb{R}$ with positive coefficients (i.e., linear functions that grow strictly monotonically in each dimension) that maps the k dimensional vectors to real numbers, we define $\mathcal{P}_f = (\Sigma, Q, q_0, \Delta, \Gamma, f \circ v)$ just like the automaton $\mathcal{P}$ from the previous subsection. The proof of Theorem 1 then trivially extends to the following theorem.

Theorem 2. *Given an MDP $\mathcal{M}$ and k lexicographic ω-regular objectives given as GFM Büchi automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$, it holds that maximising these k objectives, weighted by some $f : \mathbb{R}^k \to \mathbb{R}$ with only positive coefficients, and maximising them with the automaton $\mathcal{P}_f$ provides the same result.*

Proof. The proof of Theorem 1 merely needs to be extended by the observation that, for every run r, with value b^+ in the independent semantics, b^- in the

infimum semantics, and w_f in the weighted semantics, $f(b^+) \geqslant w_f \geqslant f(b^-)$ holds. Recalling that the same expected vectors can be obtained using the individual $\mathcal{A}_i$, $\mathcal{P}$ with independent semantics, and $\mathcal{P}$ with infimum semantics, all these maximisations provide the same result. □

Lemma 1. *Let μ be any optimal finite-memory strategy for $\mathcal{M} \times \mathcal{P}_f$. Then $(M \times \mathcal{P}_f)_\mu$ never has two transitions $t, t' \in \Gamma$ with $v(t) \neq v(t')$ in a end-component reachable from its initial state.*

Proof. Assuming such transitions t and t', the strategy can be improved by playing an adjusted strategy that mimics the strategy in the end-component (once reached), except that it only plays an accepting transition when all Boolean values that occur in this end-component in the independent semantics, are set. As this increases the expected reward, it contradicts the optimality of the end-component. □

This lemma also entails the existence of memoryless optimal strategies.

Observation. We now observe that, for carefully chosen f, optimizing the expected reward provides an optimal strategy for $\mathcal{P}$ (for both semantics). For a given MDP $\mathcal{M}$ and k GFM Büchi automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$, there is a linear function f such that an optimal pure strategy for $\mathcal{M} \times \mathcal{P}_f$ is also optimal on $\mathcal{M} \times \mathcal{P}$. (Note that $\mathcal{M} \times \mathcal{P}$ and $\mathcal{M} \times \mathcal{P}_f$ have the same states and strategies, and that Lemma 1 entails that the value for both semantics of $\mathcal{P}$ is the same.) How to choose such weights was described in Section 3.3.

3.6 From Weighted Büchi to Weighted Reachability

Generalizing the construction for GFM Büchi automata from [11,12], we replace the *fixed* payoff of $+R(\cdot) = 1$ used in the gadget (cf. Figure 2) by a *transition dependent* payoff $f\big(v(t)\big)$. The gadget is otherwise unchanged: it takes the original transition with probability ζ, and moves, with a probability of $1 - \zeta$, to a sink state T—providing a payoff of $f\big(v(t)\big)$—where the run ends. The payout on these transitions is the only reward that exists in this game, while ζ is a hyperparameter.

This has reduced the problem into a generalised reachability game, where a payout occurs only in the last step.

Theorem 3. *Given an MDP $\mathcal{M}$ and k lexicographic ω-regular objectives given as GFM Büchi automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$, and an f that satisfies the condition from the observation above, there is a $\zeta_0 < 1$ such that, for all $\zeta \in [\zeta_0, 1)$, the optimal (pure positional) strategies obtained from replacing the accepting transitions in $\mathcal{M} \times \mathcal{P}_f$ by the gadget, are optimal for $\mathcal{M} \times \mathcal{P}_f$.*

Proof. We start with expanding the observation from Lemma 1 about end-components in optimal solutions to the weighted reachability objective obtained using this gadget: in both MDPs under consideration, $\mathcal{M} \times \mathcal{P}_f$ and the variation where accepting transitions are replaced by the gadget, there cannot be two

$t, t' \in \Gamma$ with $v(t) \neq v(t')$—if there were, the expected payoff could be improved as described in the proof of Lemma 1.

This leaves the expected difference to be purely down to the part *before* reaching an end-component. Moreover, for every positional strategy, the expected value of the undiscounted payoff is between the value obtained by removing the payoff for the gadgets not in an end-component, and increasing the payoff for these gadgets to the maximal payoff. As the expected difference between these extremes goes to 0 when ζ goes to 1, for sufficiently large $\zeta < 1$, optimal strategies for weighted reachability in the model with gadgets are also optimal for the mean payoff objective of $\mathcal{M} \times \mathcal{P}_f$. □

Note that weighted reachability objectives always have optimal positional strategies. The theorems and the observation in this section show that, for suitable parameters, such an optimal positional strategy optimises the prioritised ω-regular objectives with lexicographic order.

Theorem 4. *Given an MDP $\mathcal{M}$ and k lexicographic ω-regular objectives given as GFM Büchi automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$, a suitable linear function f, and a $\zeta < 1$ sufficiently close to 1, an optimal pure positional strategies for the MDP with weighted reachability objective obtained from replacing the accepting transitions in $\mathcal{M} \times \mathcal{P}_f$ by the gadget from this subsection provide an optimal control for $\mathcal{M}$ for the individual $\mathcal{A}_i$ with lexicographic order of relevance.*

3.7 Wrapping Up Weighted Reachability

We note that weighted reachability lacks the contraction property [20,23], which makes it theoretically unsuitable for Q-learning. Learners often wrap reachability (and undiscounted payoff) into a discounted version thereof. This adds another parameter γ (the discount factor), which should be chosen significantly closer to 1 than ζ (e.g., $1 - (1 - \zeta)^2$).

4 Experimental Results

We refer to the gadget that keeps track of the accepting edges seen for each property as the tracker. We implemented the construction described in Section 3 on-the-fly, where we keep track of the states of the MDP, the automata, and the tracker, separately and compose them together at each timestep. The agent has additions actions to 'cash in' and to control any nondeterminism in the automata.

We ran Q-learning on a series of case studies that can be seen in Table 1. In Table 1, we list the name of the example, the property prioritisation order, the number of states in the MDP, the number of states in the product, the probability of satisfaction for each property under the learned strategy, and the time in seconds. We also report the values of the parameter f (which encodes the linear function for $\overrightarrow{f} = (f, 1)$ from Section 3.3 when we have two objectives, and for $\overrightarrow{f} = (f^2, f, 1)$ when we have three), the parameter ζ, the exploration rate ε, the learning rate α, the discount factor γ, the fraction under which action

Name	prop. ord.	states	prod.	prob.	time	f	ζ	ε	α	tol	ep-l	ep-n
R 4×4	1, 2, 3	16	1024	1,0,1	6.37		0.7					65k
R 4×4	2, 1, 3	16	1024	1,0,1	1.95		0.7		0.03			75k
R 5×5	1, 2, 3	25	1600	1,1,1	12.49		0.5	0.2	0.4⋆	0	250	200k
R 4×7	1, 2, 3	28	3584	1,0.75,0.25	23.18	20	0.9	0.2	0.9⋆	0	400	100k
R 4×7	2, 1, 3	28	3584	1,0.5,0	10.62	20	0.7	0.2	0.15		200	200k
Virus	1, 2	809	58248	1,0	41.43		0.97	0.5	0.2		50	150k
Virus	2, 1	809	58248	1,0.25	191.97		0.97	0.5	0.9⋆		50	700k
UAV	1, 2	11448	732672	1,1	93.57	20		0.2			40	120k
Bridges	1, . . . , 7	19	5318784	—	3.55	1	0.9		0.2			30k

Table 1: Multiobjective Q-learning results. Blank entries indicate that the default value were used. The default values are: $f = 10$, $\zeta = 0.99$, $\epsilon = 0.1$, $\alpha = 0.1$, $\gamma = 0.999$, tol= 0.01, ep-l= 30, and ep-n= 20000. Parameters that were linearly decayed to zero over training are indicated with $\star$. For Bridges the sum of the probability of satisfaction of all properties is 6. Times are in seconds.

values are considered the same during model checking of the learned strategy, the value of the episode reset timer, and the number of training episodes. Parameters were tuned manually to minimise training time. In each case, the runtime for the experiment stays below 4 minutes. Next, we discuss these case studies.

4.1 Robot

For the robot example from the introduction in Figure 1 with $\mathbf{p} = 0.5$, we have considered instances for the three scenarios discussed there, with different priorities of them. On examples R 4×4 and R 5×5 we initialise learning episodes randomly within the model to deepen exploration in order to achieve better performance. The results are in line with the analysis provided in the introduction.

Figure 3 shows an optimal strategy learned for the 4 × 7 Robot grid with property prioritisation order 2, 1, 3.

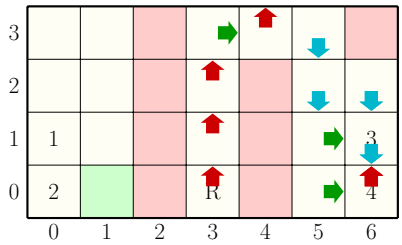

Fig. 3: Learned strategy on the 4 × 7 Robot grid with property prioritisation order 2, 1, 3. We show only reachable states and project the states in the product to the MDP. The 'cash in' action is not shown for clarity.

To avoid clutter, we show only states that are reachable under the learned strategy and do not show the 'cash in' action. Additionally, we project states in the product to the MDP. The robot starts in Row 0, Column 3 and moves up the column. Then, in Row 3, Column 3 it attempts to go across. With probability 0.5, it gets stuck in Row 3, Column 4. There are no safe actions from this field except to move north – keeping the robot in the same field. If it gets across to Row 3, Column 5, it moves down and visits the fields labeled 3 and 4.

4.2 Computer Virus

This case study (based on [18],[4]) considers the spread of a virus in a computer network. The structure of the model is sketched in Figure 4. Circles represent network nodes, and lines between them indicate network connections, between which the virus can spread. An attack gets past the firewall with probability $\mathbf{p}_{\text{detect}} = 0.5$. If past the firewall, an attack can then infect with probability $\mathbf{p}_{\text{infect}} = 0.5$. The control of the attacks is centralised. A no operation action is always available. The instance coordinating the attack has the following objectives:

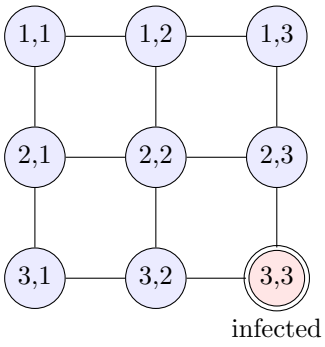

Fig. 4: Virus case study

(v_1) $\mathbf{F}s_{3,2} = 2 \bigwedge \mathbf{G}\big((s_{3,2} = 2 \wedge s_{3,1} \neq 2) \implies \mathbf{XX}s_{3,1} = 2\big)$: eventually, node $(3,2)$ gets infected; and when $(3,2)$ is infected and $(3,1)$ is not, $(3,1)$ is infected within 2 steps.

(v_2) $\mathbf{F}s_{1,1} = 2 \bigwedge \mathbf{G}(s_{2,2} \neq 2 \wedge s_{2,3} \neq 2)$: eventually node $(1,1)$ gets infected while nodes $(2,2)$ and $(2,3)$ never get infected.

When the prioritisation order is 1, 2, then the optimal strategy crosses the barrier formed by nodes $(2,2)$ and $(2,3)$ in order to infect node $(3,1)$ before node $(3,2)$. When the prioritisation order is 2, 1, then the optimal strategy respects the barrier formed by nodes $(2,2)$ and $(2,3)$, and follows the path through node $(3,1)$ to $(1,1)$. This reduces the probability of satisfying v_1 from 1 to 0.25. This problem is particularly challenging because it requires discovering a long sequence of actions and the properties interfere with each other during learning.

4.3 Human-in-the-Loop UAV Mission Planning

This model (originally from [8], [5]) considers the control of an unmanned aerial vehicle (UAV) interacting with a human operator. The UAV operates on a network of roads. In this network, waypoints (w_i) are specified as well as restricted operation zones (roz_i). In specifications, waypoints serve as places that shall be visited (once or repeatedly), while restricted operation zones shall be avoided. In our experiments, we have used the following properties.

[4] http://prismmodelchecker.org/casestudies/virus.php
[5] http://prismmodelchecker.org/casestudies/human-uav.php

(u_1) $\bigwedge_i \mathbf{G}\neg roz_i$: UAV never visits a restricted operation zone
(u_2) $\mathbf{F}w_1 \wedge \mathbf{F}w_2 \wedge \mathbf{F}w_6$: UAV eventually visits w_1, w_2, and w_6

4.4 Seven Bridges of Königsberg

As a final example, we consider the classic problem of the seven bridges of Königsberg[6], in which one seeks a path that crosses each bridge exactly once. The model is deterministic and we have 7 properties of the form

(u_i) $\mathbf{F}b_i \wedge \mathbf{G}(b_i \rightarrow \mathbf{XG} \neq b_i)$: cross bridge i exactly once.

where b_i indicates if one is on bridge i. Instead of applying a preference for each property, we set $f = 1$. In this setting, payoff is maximised when the sum of the probability of satisfaction of all properties is maximised, e.g., the agent maximises the expected number of bridges crossed exactly once. The RL agent successfully finds strategies to cross 6 bridges exactly once, the maximum number possible.

5 Conclusion

We have generalised recent work on applying model free reinforcement learning to finding optimal control for MDPs with ω-regular objectives to address the problem of controlling MDPs with multiple ω-regular objectives, and with a lexicographic order of importance: a main objective, a secondary objective, etc..

Starting with good-for-MDPs automata, the extension is surprisingly simple: it suffices to add a 'tracker' to the product of the individual automata, which memorises which of the individual Büchi conditions have occurred, and to allow the automaton to 'cash in' on these occurrences (while resetting the memory). How valuable the return is depends on the objectives, for which a good event (the passing of an individual Büchi transition) has been witnessed, where the main objective attracts a high reward and each consecutive objective obtains a small fraction of the reward assigned to the previous more important one.

Such a reward structure can be proven to work for reinforcement learning in essentially the same way as they have been proven to work for the simpler case of having a single objective (which is also a special case of our results).

This reduction is beautifully straightforward, and our experimental results show that it is also effective. This might look surprising: after all, the correctness proofs rely heavily on well chosen hyperparameters. Yet, in practice, the techniques have yet again shown to be robust: we could learn correct strategies reliably and efficiently.

References

1. de Alfaro, L.: Formal Verification of Probabilistic Systems. Ph.D. thesis, Stanford University (1998)

[6] https://en.wikipedia.org/wiki/Seven_Bridges_of_Konigsberg

2. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)
3. Bozkurt, A.K., Wang, Y., Pajic, M.: Model-free learning of safe yet effective controllers. arXiv preprint arXiv:2103.14600 (2021)
4. Bozkurt, A.K., Wang, Y., Zavlanos, M.M., Pajic, M.: Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 10349–10355 (2020). https://doi.org/10.1109/ICRA40945.2020.9196796
5. Chatterjee, K.: Markov decision processes with multiple long-run average objectives. In: Arvind, V., Prasad, S. (eds.) FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science. pp. 473–484. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
6. Chatterjee, K., Majumdar, R., Henzinger, T.A.: Markov decision processes with multiple objectives. In: Annual symposium on theoretical aspects of computer science. pp. 325–336. Springer (2006)
7. Etessami, K., Kwiatkowska, M., Vardi, M.Y., Yannakakis, M.: Multi-objective model checking of Markov decision processes. In: Grumberg, O., Huth, M. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 50–65. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
8. Feng, L., Wiltsche, C., Humphrey, L.R., Topcu, U.: Controller synthesis for autonomous systems interacting with human operators. In: Bayen, A.M., Branicky, M.S. (eds.) Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems, ICCPS 2015, Seattle, WA, USA, April 14-16, 2015. pp. 70–79. ACM (2015). https://doi.org/10.1145/2735960.2735973
9. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D., Qu, H.: Quantitative multi-objective verification for probabilistic systems. In: Abdulla, P.A., Leino, K.R.M. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 112–127. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
10. Fu, J., Topcu, U.: Probably approximately correct MDP learning and control with temporal logic constraints. In: Robotics: Science and Systems (Jul 2014)
11. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Omega-regular objectives in model-free reinforcement learning. In: Tools and Algorithms for the Construction and Analysis of Systems. pp. 395–412 (2019), LNCS 11427
12. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Good-for-mdps automata for probabilistic analysis and reinforcement learning. In: Tools and Algorithms for the Construction and Analysis of Systems (2020)
13. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Faithful and effective reward schemes for model-free reinforcement learning of omega-regular objectives. In: Hung, D.V., Sokolsky, O. (eds.) Automated Technology for Verification and Analysis - 18th International Symposium, ATVA 2020, Hanoi, Vietnam, October 19-23, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12302, pp. 108–124. Springer (2020). https://doi.org/10.1007/978-3-030-59152-6_6
14. Hasanbeig, M., Abate, A., Kroening, D.: Logically-correct reinforcement learning. CoRR **abs/1801.08099** (2018), http://arxiv.org/abs/1801.08099
15. Hasanbeig, M., Abate, A., Kroening, D.: Certified reinforcement learning with logic guidance. arXiv e-prints **arXiv:1902.00778** (Feb 2019)
16. Kretínský, J., Pérez, G.A., Raskin, J.: Learning-based mean-payoff optimization in an unknown MDP under omega-regular constraints. In: Schewe, S., Zhang, L. (eds.) 29th International Conference on Concurrency Theory, CONCUR 2018, September 4-7, 2018, Beijing, China. LIPIcs, vol. 118, pp. 8:1–8:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018). https://doi.org/10.4230/LIPIcs.CONCUR.2018.8

17. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Computer Aided Verification (CAV). pp. 585–591 (Jul 2011), LNCS 6806
18. Kwiatkowska, M., Norman, G., Parker, D., Vigliotti, M.: Probabilistic mobile ambients. Theoretical Computer Science **410**(12–13), 1272–1303 (2009)
19. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)
20. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley (1994)
21. Sadigh, D., Kim, E., Coogan, S., Sastry, S.S., Seshia, S.A.: A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications. In: CDC. pp. 1091–1096 (Dec 2014)
22. Strehl, A.L., Li, L., Wiewiora, E., Langford, J., Littman, M.L.: PAC model-free reinforcement learning. In: International Conference on Machine Learning, ICML. pp. 881–888 (2006)
23. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, second edn. (2018)
24. Watkins, C.J., Dayan, P.: Q-learning. Machine learning **8**(3-4), 279–292 (1992)